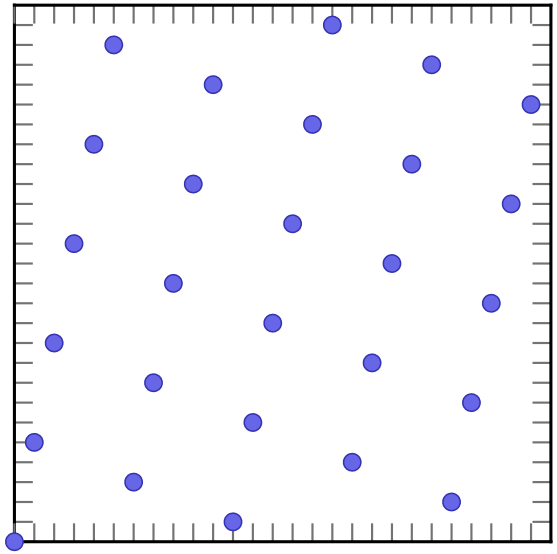
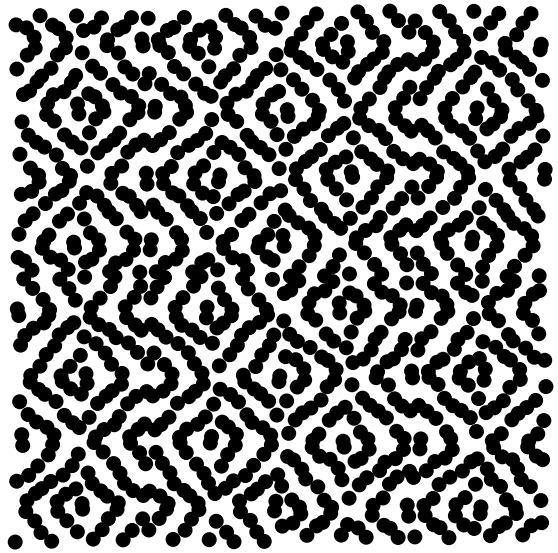
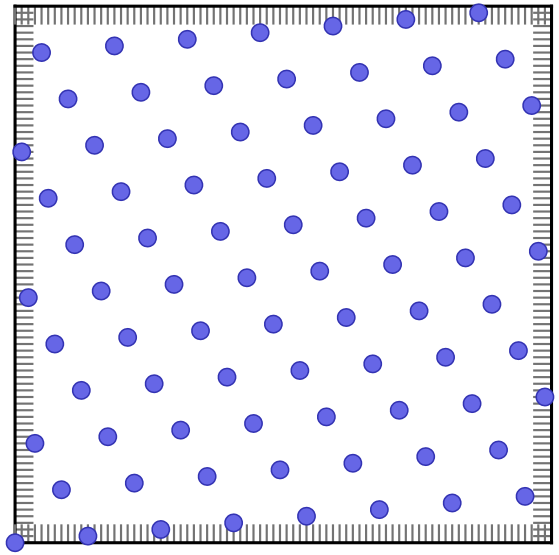
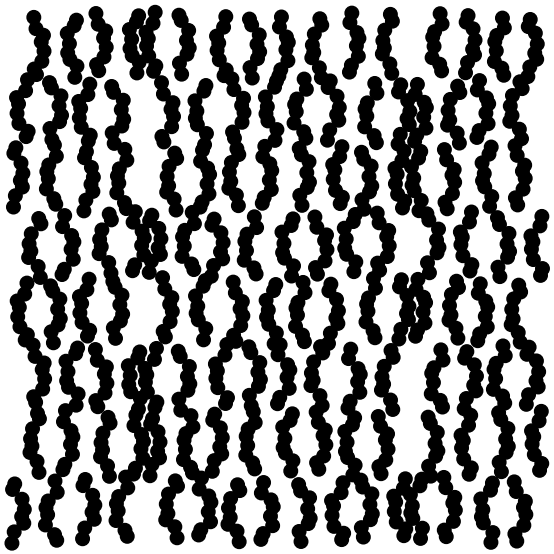
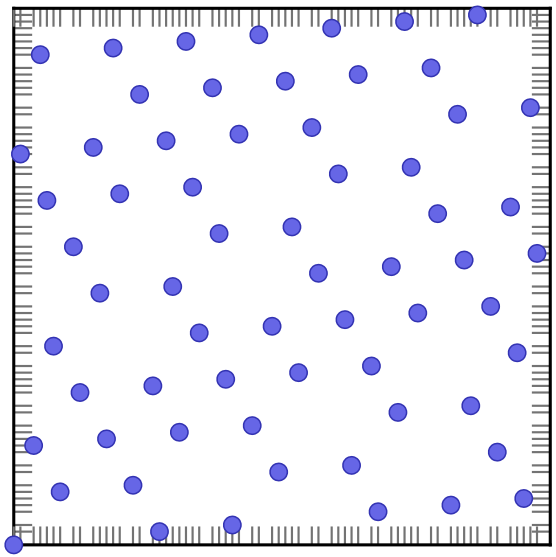
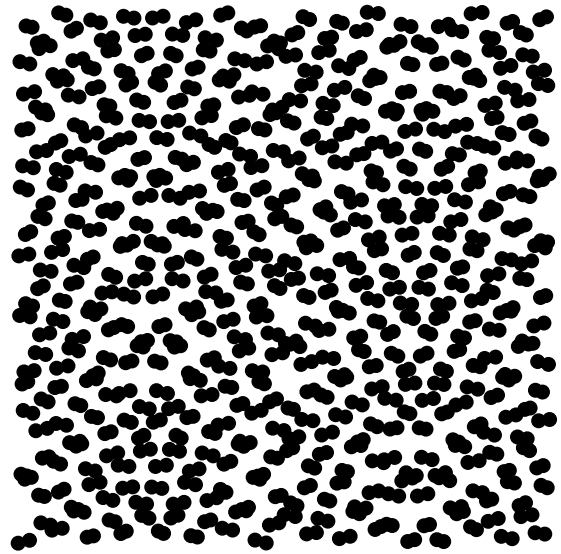




Lattice rules and beyond. . .

— Dirk Nuyens —

*Numerical Analysis and Applied Mathematics, Department of Computer Science, KU Leuven, Belgium • MCQMC2016, Stanford, August 14–19, 2016

High-dimensional integrals

Task: Approximate a d-dimensional integral

$$\mathbb{E}[f] = I(f) := \int_{\mathbb{R}^d} f(\boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \int_{[0,1]^d} f(P^{-1}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{x}.$$

Method: An N-point cubature/quadrature method

$$Q(f) = Q_N(f) = Q(f; \{(w_k, \boldsymbol{x}_k)\}_{k=1}^N) := \sum_{k=1}^N w_k\, f(\boldsymbol{x}_k).$$

(This talk: Using functional analysis and number theoretic uniform point sets. To tackle integration, approximation and "other" high dimensional problems.)

Applications: random fields, parametrised PDEs, financial engineering, Bayesian integrals, uncertainty quantification,...

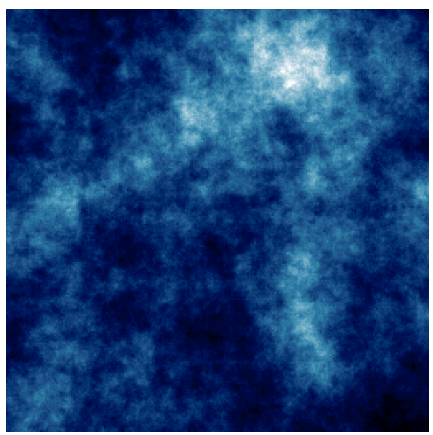
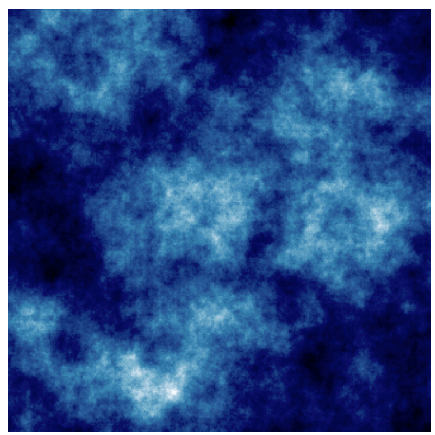
Example: random fields / parametrised PDEs ($d = \infty$)

Parametric representation (e.g., Karhunen–Loève expansion)

$$a(\boldsymbol{x}, \boldsymbol{y}) = a_0(\boldsymbol{x}) + \sum_{j=1}^{\infty} y_j \, \psi_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in D, \quad \boldsymbol{y} \in [-\tfrac{1}{2}, \tfrac{1}{2}]^{\mathbb{N}},$$

by sample variables y_j. Use in porous flow using Darcy's law:

$$q(\boldsymbol{x}, \boldsymbol{y}) + a(\boldsymbol{x}, \boldsymbol{y}) \nabla p(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}),$$
$$\nabla \cdot q(\boldsymbol{x}, \boldsymbol{y}) = 0.$$

See, e.g., Barth, Charrier, Cliffe, Dick, Gantner, Giles, Graham, Haji-Ali, Harbrecht, Kuo, Le Gia, N., Nichols, Nobile, Peters, Robbe, Scheichl, Schwab, Siebenmorgen, Sloan, Teckentrup, Tempone, Ullmann, Vandewalle, Zollinger, von Schwerin, ...

Example: option pricing (d = hundreds, thousands, ∞)

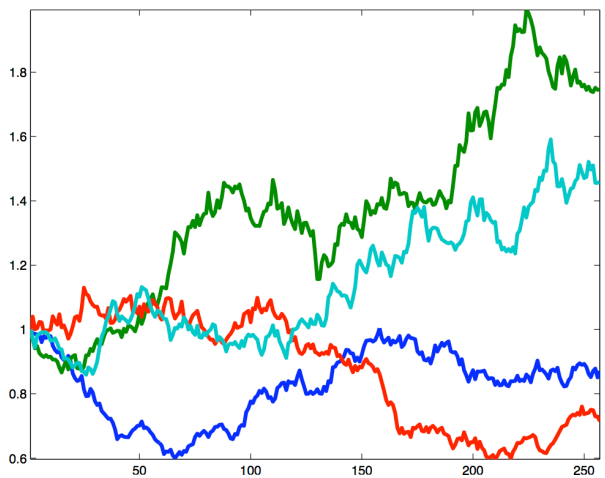
Simulation of SDE

$$\mathrm{d}X(t) = a(X(t))\,\mathrm{d}t + b(X(t))\,\mathrm{d}W(t), \quad X(0) = X_0, \quad 0 \le t \le T,$$

using Euler–Maruyama, $\widehat{X}_0 = X_0$,

$$\widehat{X}_{i+1} = \widehat{X}_i + a(\widehat{X}_i)\,h + b(\widehat{X}_i)\,\sqrt{h}\,Z_i, \quad i = 1, \ldots, n-1, \quad h = T/n,$$

with Z_i sampled from standard normal distribution.



See, e.g., Achtsis, Baldeaux, Boyle, Cools, Gerstner, Giles, Glasserman, Griebel, Holtz, Imai, Irrgeher, Joshi, Kucherenko, Kuo, L'Écuyer, Larcher, Lemieux, Leobacher, Lin, N., Niu, Ökten, Pages, Platen, Sloan, Staum, Szölgyenyi, Tan, Tezuka, Tichy, Traub, Tuffin, Wang, Waterhouse, ...

Example: option pricing (d = hundreds, thousands, ∞)

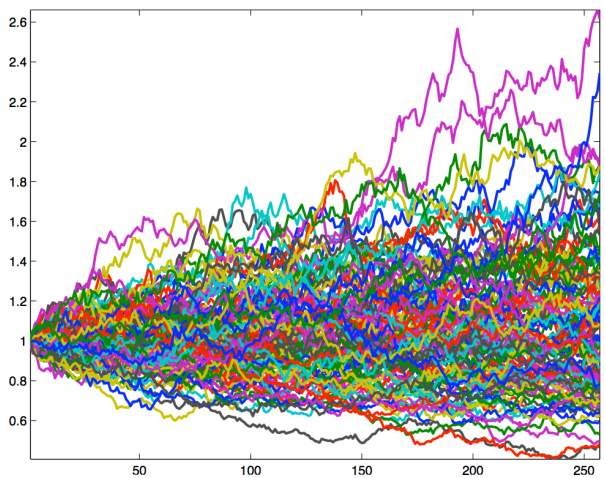
Simulation of SDE

$$\mathrm{d}X(t) = a(X(t))\,\mathrm{d}t + b(X(t))\,\mathrm{d}W(t), \quad X(0) = X_0, \quad 0 \le t \le T,$$

using Euler–Maruyama, $\widehat{X}_0 = X_0$,

$$\widehat{X}_{i+1} = \widehat{X}_i + a(\widehat{X}_i)\,h + b(\widehat{X}_i)\,\sqrt{h}\,Z_i, \quad i = 1, \ldots, n-1, \quad h = T/n,$$

with Z_i sampled from standard normal distribution.



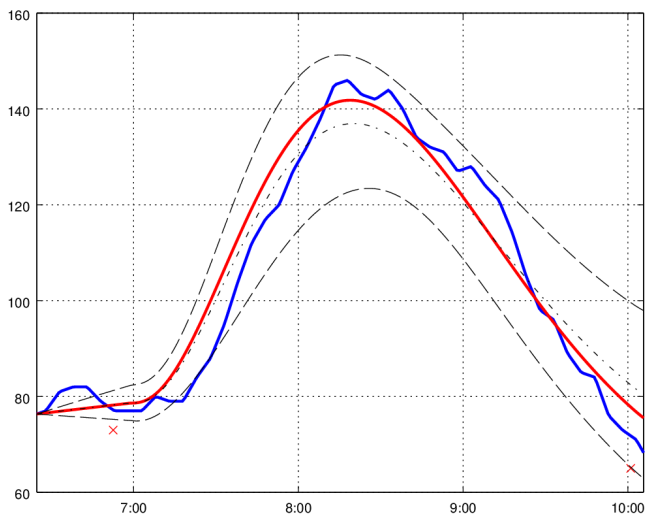
See, e.g., Achtsis, Baldeaux, Boyle, Cools, Gerstner, Giles, Glasserman, Griebel, Holtz, Imai, Irrgeher, Joshi, Kucherenko, Kuo, L'Écuyer, Larcher, Lemieux, Leobacher, Lin, N., Niu, Ökten, Pages, Platen, Sloan, Staum, Szölgyenyi, Tan, Tezuka, Tichy, Traub, Tuffin, Wang, Waterhouse, ...

Example: Bayesian integrals (d = hundreds, thousands, ∞)

Simulation of insulin-glucose model

$$\mathrm{d}G(t)/\mathrm{d}t = -\lambda\big(G(t) - G_b\big) - \beta X(t) G(t) + R_a(t)$$
$$\mathrm{d}X(t)/\mathrm{d}t = -\mu X(t) + \mu\big(I(t) - I_b\big)$$

to infer parameters and quantify input uncertainty given noisy measurement $G^{\eta}_{\mathrm{ref}}(t)$ and uncertain input data $R_a(t)$.



Using evaluation of the integral point-of-view: see, e.g., Dick, Gantner, Le Gia, N., Scheichl, Schillings, Schwab, Stuart, Teckentrup, ...

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?
  Assume a product of one-dimensional rules with the minimum number of points per dimension:
  - $m = 1$? $\Rightarrow N = 1^d = 1$, initial approximation?

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?
  Assume a product of one-dimensional rules with the minimum number of points per dimension:
  - $m = 1?$ $\Rightarrow N = 1^d = 1$, initial approximation?
  - $m = 2?$ $\Rightarrow N = 2^d$, next approximation?

  So, what if $d = 100$, $N = 2^{100}$?

$$2^{100} = 1267650600228229401496703205376 \approx 10^{30}.$$

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?
  Assume a product of one-dimensional rules with the minimum number of points per dimension:
  - $m = 1?$ $\Rightarrow N = 1^d = 1$, initial approximation?
  - $m = 2?$ $\Rightarrow N = 2^d$, next approximation?

  So, what if $d = 100$, $N = 2^{100}$?

$$2^{100} = 1267650600228229401496703205376 \approx 10^{30}.$$

That is called a quintillion...

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?
  Assume a product of one-dimensional rules with the minimum number of points per dimension:
  - $m = 1?$ $\Rightarrow N = 1^d = 1$, initial approximation?
  - $m = 2?$ $\Rightarrow N = 2^d$, next approximation?

  So, what if $d = 100$, $N = 2^{100}$?

$$2^{100} = 1267650600228229401496703205376 \approx 10^{30}.$$

That is called a quintillion...

At 1 function evaluation per μs that is $10^{-6} \times 10^{30} = 10^{24}$ sec.

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?
  Assume a product of one-dimensional rules with the minimum number of points per dimension:
  - $m = 1?$ $\Rightarrow N = 1^d = 1$, initial approximation?
  - $m = 2?$ $\Rightarrow N = 2^d$, next approximation?

  So, what if $d = 100$, $N = 2^{100}$?

$$2^{100} = 1267650600228229401496703205376 \approx 10^{30}.$$

That is called a quintillion...
At 1 function evaluation per μs that is $10^{-6} \times 10^{30} = 10^{24}$ sec.
And the age of the universe is... roughly 10^{17} sec!

Product rules

"The curse by construction"

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules?
  Assume a product of one-dimensional rules with the minimum number of points per dimension:
  - $m = 1?$ $\Rightarrow N = 1^d = 1$, initial approximation?
  - $m = 2?$ $\Rightarrow N = 2^d$, next approximation?

  So, what if $d = 100$, $N = 2^{100}$?

$$2^{100} = 1267650600228229401496703205376 \approx 10^{30}.$$

That is called a quintillion...
At 1 function evaluation per μs that is $10^{-6} \times 10^{30} = 10^{24}$ sec.
And the age of the universe is... roughly 10^{17} sec!

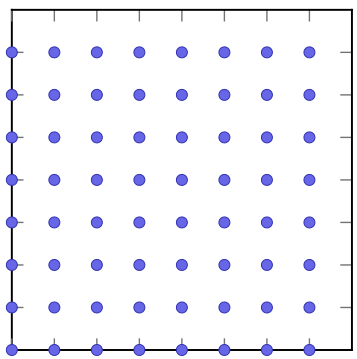
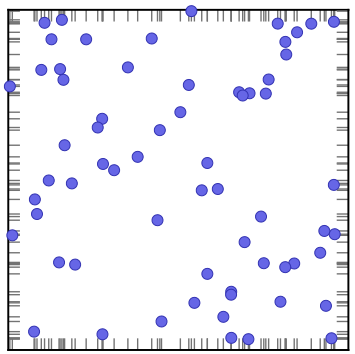
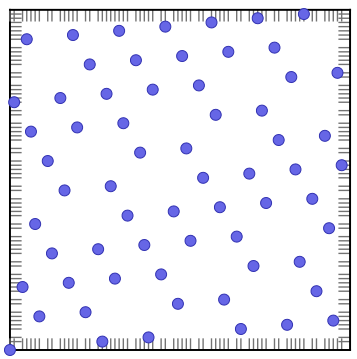
THE CURSE OF DIMENSIONALITY!

(Could try luck with "sparse" grid method; but not $d = 100$ out of the box, same issue.)
(An appropriate definition for the curse and intractability follows.)

Monte Carlo type methods: $\frac{1}{N}\sum_{k=1}^{N} f(\boldsymbol{x}_k)$

What kind of cubature/quadrature method to use for d large?

- A product of classical quadrature rules? 💣
  → $N = m^d$ ⇒ The curse "by construction"!
- The plain Monte Carlo method: $\boldsymbol{x}_k \sim U[0,1)^d$.
  → Free to choose N.
- Quasi-Monte Carlo methods: deterministic points using some algebraic structure.
  → Free to choose N.

grid	MC	QMC
		
$N = m^d$	N free	N free
error $= O(N^{-r/d})$	std $= O(N^{-1/2})$	error $= O(N^{-1}), \ldots$

Lattice rules rule! Lattice rules rule! Lattice rules rule!

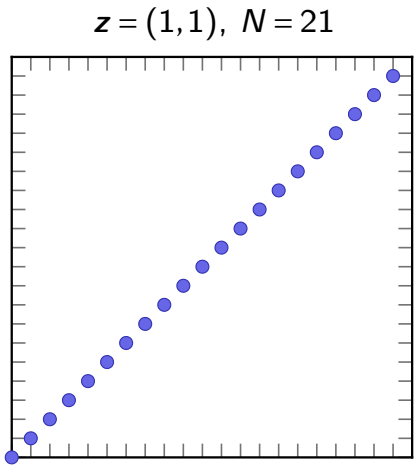
Rank-1 lattice rule: given N and integer generating vector $\boldsymbol{z}$:

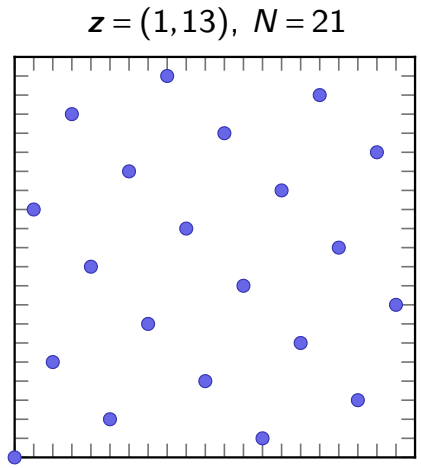
$$\boldsymbol{x}_k = \left\{ \frac{\boldsymbol{z}k}{N} \right\} := \frac{\boldsymbol{z}k}{N} \bmod 1 = \frac{\boldsymbol{z}k \bmod N}{N}, \qquad k = 0, \ldots, N-1.$$

Classical references: Niederreiter (1992), Sloan & Joe (1994).

"Good" lattice rules require "good" generating vectors $\boldsymbol{z} \in \mathbb{Z}^d$.

Obviously $\boldsymbol{z} = (1, \ldots, 1)$ will only have diagonal points in $[0,1]^d$.





Lattice rules rule! Lattice rules rule! Lattice rules rule!

Rank-1 lattice rule: given N and integer generating vector $\boldsymbol{z}$:

$$\boldsymbol{x}_k = \left\{ \frac{\boldsymbol{z}k}{N} \right\} := \frac{\boldsymbol{z}k}{N} \bmod 1 = \frac{\boldsymbol{z}k \bmod N}{N}, \qquad k = 0, \ldots, N-1.$$

Classical references: Niederreiter (1992), Sloan & Joe (1994).

"Good" lattice rules require "good" generating vectors $\boldsymbol{z} \in \mathbb{Z}^d$.

Obviously $\boldsymbol{z} = (1, \ldots, 1)$ will only have diagonal points in $[0,1]^d$.

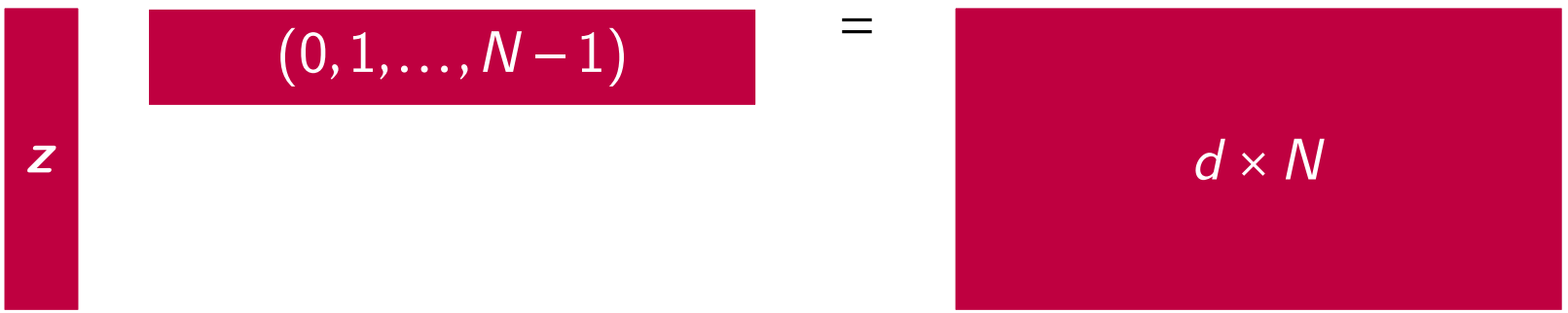
How to find "good" rules? (More later in the talk.)

→ 2D: Fibonacci lattice rules: $\boldsymbol{z} = (1, F_{k-1})$, $N = F_k$.

→ Component-by-component construction, see, e.g., Dick, Joe, Korobov, Kritzer, Kuo, N., Pillichshammer, Sloan, Suryanarayana, Weimar, ...

→ Fast component-by-component construction, see, e.g, Cools, Dick, Korobov, Kritzer, Laimer, Leobacher, Kuo, Le Gia, N., Matthysen, Pillichshammer, Schwab, ...

Visual mind-image (vectorization)



One-liners in Matlab/Octave, Python, ...

So assume you are given a "good" $\boldsymbol{z}$ for your choice of N, then

$$Q_N(f;\boldsymbol{z},N) = \frac{1}{N}\sum_{k=0}^{N-1} f\left(\frac{\boldsymbol{z}k \bmod N}{N}\right), \quad \text{and take eg } f(\boldsymbol{x}) = \left|\sum_{j=1}^{d} \mathrm{e}^{2\pi \mathrm{i} x_j}\right|^r.$$

(Example $I(f)$ is expected distance of rth moment of distance travelled by d-step random walk in the plane, see, Borwein, N., Straub, Wan (2011).)

One-liners in Matlab/Octave, Python, ...

So assume you are given a "good" $\boldsymbol{z}$ for your choice of N, then

$$Q_N(f;\boldsymbol{z},N) = \frac{1}{N}\sum_{k=0}^{N-1} f\left(\frac{\boldsymbol{z}k \bmod N}{N}\right), \quad \text{and take eg } f(\boldsymbol{x}) = \left|\sum_{j=1}^{d} \mathrm{e}^{2\pi \mathrm{i} x_j}\right|^r.$$

(Example $I(f)$ is expected distance of rth moment of distance travelled by d-step random walk in the plane, see, Borwein, N., Straub, Wan (2011).) (Copy paste does not work, watch '-' and '*'. Sorry.)

```
z = [1; 55]; N = 89; % points as [d x N] (Fortran), dim=1,2
f = @(r, x) abs( sum( exp(2*pi*1i*x), 1 ) ).^r;
% one-liner:
Q = mean( f(1, mod(z*(0:N-1), N)/N ) )
```

One-liners in Matlab/Octave, Python, ...

So assume you are given a "good" $\boldsymbol{z}$ for your choice of N, then

$$Q_N(f;\boldsymbol{z},N) = \frac{1}{N}\sum_{k=0}^{N-1} f\left(\frac{\boldsymbol{z}k \bmod N}{N}\right), \quad \text{and take eg } f(\boldsymbol{x}) = \left|\sum_{j=1}^{d} \mathrm{e}^{2\pi \mathrm{i} x_j}\right|^r.$$

(Example $I(f)$ is expected distance of rth moment of distance travelled by d-step random walk in the plane, see, Borwein, N., Straub, Wan (2011).) (Copy paste does not work, watch '-' and '*'. Sorry.)

```
z = [1; 55]; N = 89; % points as [d x N] (Fortran), dim=1,2
f = @(r, x) abs( sum( exp(2*pi*1i*x), 1 ) ).^r;
% one-liner:
Q = mean( f(1, mod(z*(0:N-1), N)/N ) )
```

```
from numpy import *; # using Numpy
z = [1, 55]; N = 89; # points as [N x d] (C), axis=0,1
f = lambda r, x: abs( sum( exp(2*pi*1j*x), axis=1 ) )**r
# one-liner:
Q = mean( f(1, (outer(range(N), z) % N)/float(N) ) )
```

Fixed N is boring, so, . . . , in come lattice sequences

- Let $\varphi_b(k)$ be the radical inverse function in base b:

$$\varphi_b : \mathbb{N}_0 \to [0,1) : (\cdots k_3 k_2 k_1 k_0)_b \mapsto (0.k_0 k_1 k_2 k_3 \cdots)_b.$$

- Hickernell & Niederreiter (2003), Hickernell, Hong, L'Écuyer, Lemieux (2000),

$$\boldsymbol{x}_k = \boldsymbol{z}\,\varphi_b(k) \bmod 1, \qquad k = 0,1,\ldots,$$

equivalently, for some m,

$$\boldsymbol{x}_k = (\boldsymbol{z}\,\mathrm{rev}_{b,m}(k) \bmod b^m)/b^m, \qquad k = 0,\ldots,b^m-1,$$

$\mathrm{rev}_{b,m}(k) = [\varphi_b(k) b^m]$ reverses the digits in an m-digit word.

→ Fast CBC construction of rules good for a range of m: Cools, Kuo, N. (2016), Dick, Pillichshammer, Waterhouse (2008).
→ For $b = 2$, bitreverse can be done in $\log_2(m)$ (bithacks).

An alternative sequence construction

- Typical trick to generate an $(s+1)$-dimensional digital net with b^m points from an s-dimensional sequence: add $x_0 = k/b^m$, $k = 0, \ldots, b^m - 1$, as the first dimension.

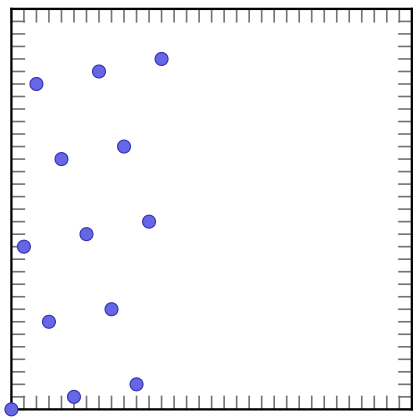
An alternative sequence construction

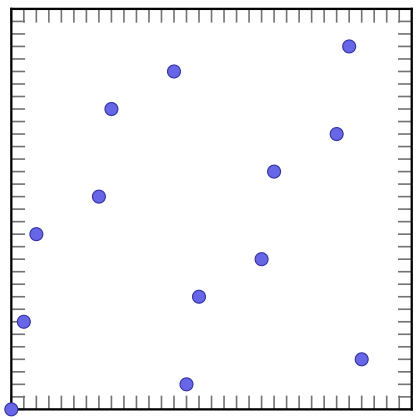
- Typical trick to generate an $(s+1)$-dimensional digital net with b^m points from an s-dimensional sequence: add $x_0 = k/b^m$, $k = 0, \ldots, b^m - 1$, as the first dimension.
- Idea from Korobov (1961): take s-dimensional lattice rule with $z_1 = 1$, drop first dimension to obtain $(s-1)$-dimensional lattice sequence. Ok for $O(N^{-1})$, see Hickernell, Kuo, Kritzer & N. (2011).

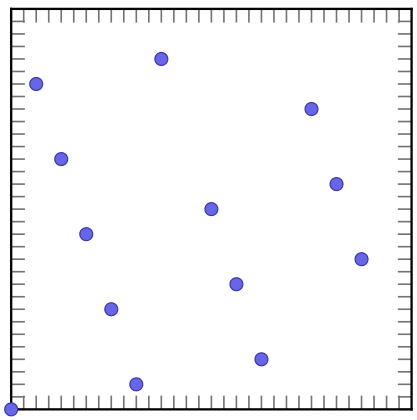
An alternative sequence construction

- Typical trick to generate an $(s+1)$-dimensional digital net with b^m points from an s-dimensional sequence: add $x_0 = k/b^m$, $k = 0,\dots,b^m-1$, as the first dimension.
- Idea from Korobov (1961): take s-dimensional lattice rule with $z_1 = 1$, drop first dimension to obtain $(s-1)$-dimensional lattice sequence. Ok for $O(N^{-1})$, see Hickernell, Kuo, Kritzer & N. (2011).

"First" 13 points of a base 2 lattice sequence optimized over $2^1,\dots,2^5$ with $z = (1,13,27,\dots)$:


Initial part of lattice


Korobov-seq


Radical-inverse-seq

Korobov trick for lattice sequence

```
% For a rule with z(1)=1 and length(z) >= d+1
acc = sum( f( mod( z(2:d+1) * (0:n1-1) , N ) / N )
Q1 = acc / n1
acc = acc + sum( f( mod( z(2:d+1) * (n1:n2-1) , N ) / N )
Q2 = acc / n2
...
```

Korobov trick for lattice sequence

```
% For a rule with z(1)=1 and length(z) >= d+1
acc = sum( f( mod( z(2:d+1) * (0:n1-1) , N ) / N )
Q1 = acc / n1
acc = acc + sum( f( mod( z(2:d+1) * (n1:n2-1) , N ) / N )
Q2 = acc / n2
...
```

Note: Although the above is a cool trick, we provide a Matlab/Octave point generator which can be used like `rand` and uses the radical-inverse method (using the bithacks bitreverse):

```
latticeseq_b2('init0', z, N);
acc = sum( f( latticeseq_b2(d, n1) )
Q1 = acc / n1
acc = acc + sum( f( latticeseq_b2(d, n2) )
Q2 = acc / n2
...
```

Randomly shifted lattice rules

To obtain an error estimator: take i.i.d. random shifts of the lattice

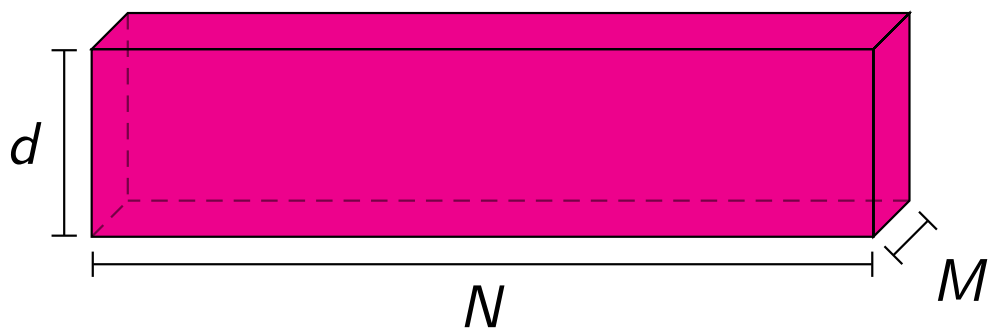
$$\boldsymbol{x}_k^{(i)} = \left(\boldsymbol{x}_k + \boldsymbol{\Delta}^{(i)}\right) \bmod 1 = \frac{\left(\boldsymbol{z}k + N\boldsymbol{\Delta}^{(i)}\right) \bmod N}{N}, \quad i = 1, \ldots, M,$$

to obtain M independent estimates $Q_N^{(1)}, \ldots, Q_N^{(M)}$. Then

$$\overline{Q}_{M\times N} = \frac{1}{M}\sum_{i=1}^{M} Q_N^{(i)}, \quad \mathrm{std}(\overline{Q}_{M\times N}(f)) = \frac{\mathrm{std}(Q_N^{(1)}, \ldots, Q_N^{(M)})}{\sqrt{M}}.$$

Singleton expansion in Matlab/Octave to generate the points $[d \times N \times M]$:

```
mod(bsxfun(@plus, reshape(z*(0:N-1), d, N, 1),
                  reshape(N*shifts, d, 1, M)), N) / N
```



How to measure deterministic algorithms?

- Worst-case error for approximating $I(f)$ by $Q_N(f)$ for $f \in \mathcal{F}_d$:

$$e(Q_N;\mathcal{F}_d) := \sup_{\substack{f\in\mathcal{F}_d \\ \|f\|_{\mathcal{F}_d}\le 1}} \big|I(f)-Q_N(f)\big| \le \text{upper bound for } Q_N.$$

(And the RMS expectation over random shifts in case of a randomly shifted rule.)

- Best possible error using N function values (benchmark):

$$e(N;\mathcal{F}_d) := \inf_{Q_N:\{(w_k,\boldsymbol{x}_k)\}_{k=1}^N} e(Q_N;\mathcal{F}_d) \ge \text{lower bound for any } Q_N$$

$$= \textit{error of best algorithm using } N \textit{ function evaluations}.$$

- Information complexity: the minimal number of function values needed to reach error at most ϵ:

$$n(\epsilon;\mathcal{F}_d) := \min\big\{N : \exists Q_N \text{ for which } e(Q_N;\mathcal{F}_d)\le\epsilon\big\}$$

$$= \textit{number of function evaluations of best algorithm}.$$

See a multitude of references, e.g., Novak (2016) or the Novak–Woźniakowski trilogy (2008,2010,2012), ...

The curse of dimensionality & types of tractability

Tractability started by Woźniakowski (1994) and since then vastly expanding...

- The curse of dimensionality is now defined as needing an exponential number of function values in d to reach an error $\epsilon \leq \epsilon_0$:

$$n(\epsilon; \mathscr{F}_d) \geq c\,(1+\gamma)^d, \qquad \text{for some } c, \gamma, \epsilon_0 > 0.$$

- A problem is called (weakly) tractable if

$$\lim_{\epsilon^{-1}+d \to \infty} \frac{\ln n(\epsilon, d)}{\epsilon^{-1} + d} = 0,$$

  and intractable otherwise.
- Different types, e.g., polynomial tractability

$$n(\epsilon; \mathscr{F}_d) \leq c\,\epsilon^{-p}\,d^q, \qquad \text{for some } c, p, q \geq 0.$$

See a multitude of references, in particular the Novak–Woźniakowski trilogy (2008,2010,2012), ...

The curse might always be there...

Assume $f \in \mathscr{F}_d$ when

$$\|f\|_{\mathscr{F}_d} := \max_{\boldsymbol{x},\boldsymbol{y}\in[0,1]^d} \frac{|f(\boldsymbol{x})-f(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|_\infty} \quad < \quad \infty,$$

then (Maung Zho Newn and Sharygin, 1971)

$$e(N,\mathscr{F}_d) = \frac{d}{2d+2} N^{-1/d}.$$

See Novak's (2016) review in MCQMC2014:

Not just avoid the "curse by construction", but also

- rate independent of d $\Rightarrow$ "mixed dominating smoothness".
- constant $C_{d,\alpha}$ independent of d $\Rightarrow$ "weighted spaces".

Aims

- **Mixed dominating smoothness spaces:** Move from typical Sobolev norm with $\|D^{\boldsymbol{\tau}} f\|$ bounded for $\tau_1 + \cdots + \tau_d \leq \alpha$, which gives $O(N^{-\alpha/d})$ to $(\tau_1, \ldots, \tau_d) \leq \alpha$ which gives $\sim O(N^{-\alpha})$.
- **Dimension-independent error bounds:** Switch to weighted spaces: not all combinations of variables are as important. Denote the importance of the variables in $\mathfrak{u} \subseteq \{1, \ldots, d\}$ by $\gamma_{\mathfrak{u}}$.

Mixed dominating smoothness: Novak, Sickel, Temlyakov, Ullrich ×2, ...
Weights: Hickernell (1998), Sloan & Woźniakowski (1998), Novak–Woźniakowski trilogy (2008,2010,2012), ...

Spaces based on series representations & Koksma–Hlawka

Assume an L_2-ONB $\{\phi_h\}_h$, with $\phi_0 = 1$, $Q_N(1) = 1$, and abs. summ.

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \widehat{f}(\boldsymbol{h})\,\phi_{\boldsymbol{h}}(\boldsymbol{x}), \qquad \text{with} \quad \widehat{f}(\boldsymbol{h}) := \int_{[0,1]^d} f(\boldsymbol{x})\,\overline{\phi_{\boldsymbol{h}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x},$$

Spaces based on series representations & Koksma–Hlawka

Assume an L_2-ONB $\{\phi_{\boldsymbol{h}}\}_{\boldsymbol{h}}$, with $\phi_0 = 1$, $Q_N(1) = 1$, and abs. summ.

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \widehat{f}(\boldsymbol{h})\,\phi_{\boldsymbol{h}}(\boldsymbol{x}), \qquad \text{with} \qquad \widehat{f}(\boldsymbol{h}) := \int_{[0,1]^d} f(\boldsymbol{x})\,\overline{\phi_{\boldsymbol{h}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x},$$

then, for $r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h}) > 0$ an "increasing" function,

$$\begin{aligned} \big|I(f) - Q_N(f)\big| &= \left|\sum_{\boldsymbol{h}\neq\boldsymbol{0}} \widehat{f}(\boldsymbol{h})\, Q_N(\phi_{\boldsymbol{h}})\, r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})\, r_{\alpha,\boldsymbol{\gamma}}^{-1}(\boldsymbol{h})\right| \\ &\le \left(\sum_{\boldsymbol{h}} \big|\widehat{f}(\boldsymbol{h})\big|^p\, r_{\alpha,\boldsymbol{\gamma}}^{p}(\boldsymbol{h})\right)^{1/p} \left(\sum_{\boldsymbol{h}\neq\boldsymbol{0}} \big|Q_N(\phi_{\boldsymbol{h}})\big|^q\, r_{\alpha,\boldsymbol{\gamma}}^{-q}(\boldsymbol{h})\right)^{1/q} \\ &\qquad\qquad \text{norm} \qquad\qquad \times \qquad\qquad \text{worst-case error}. \end{aligned}$$

For $1 < p \le \infty$ and nice choices of $\phi_{\boldsymbol{h}}$, Q_N and $r_{\alpha,\boldsymbol{\gamma}}$ we can find a "worst-case" representer $\xi(\boldsymbol{x})$ for which

$$\big|Q_N(\xi) - I(\xi)\big|^{1/q} = e(Q_N, \mathscr{F}_d),$$

independent of the particular Q_N, e.g., Fourier series and lattice rules, Walsh series and digital nets, see N. (2014) and Hickernell (1998a,b).

Choice I

So we have (Let $\mathfrak{u}(\boldsymbol{h})$ be the support of $\boldsymbol{h}$.)

- choice of $r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})$, controls the (decaying) importance of $\phi_{\boldsymbol{h}}$:
  - polynomial decay, e.g., $r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h}) = \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} (1+h_j)^{\alpha}$,
  - exponential decay, e.g., $r_{\boldsymbol{a},\boldsymbol{b},\boldsymbol{\gamma}}(\boldsymbol{h}) = \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} \exp(a_j h_j^{b_j})$,
- choice of $\boldsymbol{\gamma} = \{\gamma_{\mathfrak{u}}\}_{\mathfrak{u} \subseteq \{1,\ldots,d\}}$:
  - product weights: $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_{\{j\}}$,
  - order-dependent weights: $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|}$,
  - POD weights: $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_{\{j\}}$,
  - ...,
- choice of $\phi_{\boldsymbol{h}}$,
- choice of Q_N, i.e., $\{(w_k, \boldsymbol{x}_k)\}_{k=1}^{N}$.

Weights: Sloan, Woźniakowski (1998), Dick, Sloan, Wang, Woźniakowski (2004), Kuo, Schwab, Sloan (2012), Dick, Kuo, Le Gia, N., Schwab (2014), ...
Cos: Dick, N., Pillichshammer (2013), Suryanarayana, N., Cools (2015), Potts, Volkmer (2015), Cools, Kuo, N., Suryanarayana (2016), ...
Exp: Dick, Larcher, Pillichshammer, Woźniakowski (2011), Kritzer, Pillichshammer, Woźniakowski (2014), Irrgeher, Kritzer, Leobacher, Pillichshammer. (2015), Nguyen, N. (201x), Nguyen, N., Cools (201y), ...

Choice II

So we have

- choice of $r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})$, controls the (decaying) importance of $\phi_{\boldsymbol{h}}$,
- choice of $\boldsymbol{\gamma} = \{\gamma_{\mathfrak{u}}\}_{\mathfrak{u}\subseteq\{1,\ldots,d\}}$,
- choice of $\phi_{\boldsymbol{h}}$:
  - Fourier basis,
  - Walsh basis,
  - Cosine basis,
- choice of Q_N, i.e., $\{(w_k, \boldsymbol{x}_k)\}_{k=1}^N$:
  - Lattice rules/sequences,
  - Digital nets/sequences, polynomial lattice rules,
  - Tent-transformed lattice rules/sequences.

Weights: Sloan, Woźniakowski (1998), Dick, Sloan, Wang, Woźniakowski (2004), Kuo, Schwab, Sloan (2012), Dick, Kuo, Le Gia, N., Schwab (2014), ...
Cos: Dick, N., Pillichshammer (2013), Suryanarayana, N., Cools (2015), Potts, Volkmer (2015), Cools, Kuo, N., Suryanarayana (2016), ...
Exp: Dick, Larcher, Pillichshammer, Woźniakowski (2011), Kritzer, Pillichshammer, Woźniakowski (2014), Irrgeher, Kritzer, Leobacher, Pillichshammer. (2015), Nguyen, N. (201x), Nguyen, N., Cools (201y), ...

Reproducing kernel Hilbert spaces, $p = q = 2$

Given a one-dimensional reproducing kernel $K(x,y) = \overline{K(y,x)}$.
Suppose $\mathscr{H}(K)$ is separable: $\mathscr{H}(K) = \operatorname{span}\{\phi_h\}_h$ and $\phi_0 = 1$.
Determine the eigenvalues and eigenfunctions, and assume $\lambda_0 = 1$,

$$\int_{[0,1]} \phi(x)\,\overline{K(x,y)}\,\mathrm{d}x = \lambda\,\phi(y).$$

Then

$$K(x,y) = \sum_h \frac{\phi_h(x)}{\sqrt{\lambda_h}}\,\frac{\overline{\phi_h(y)}}{\sqrt{\lambda_h}} = \sum_h \frac{\phi_h(x)}{\|\phi_h\|_{L_2}}\,\frac{\overline{\phi_h(y)}}{\|\phi_h\|_{L_2}},$$

the ϕ_h are L_2-orthogonal, with $\|\phi_h\|_{L_2} = \sqrt{\lambda_h}$ and $\|\phi_h\|_{\mathscr{H}} = 1$, with

$$\langle f,g\rangle_{\mathscr{H}} = \sum_h \lambda_h\,\widehat{f}(h)\,\overline{\widehat{g}(h)}, \qquad \|f\|^2_{\mathscr{H}} = \sum_h \lambda_h\,\bigl|\widehat{f}(h)\bigr|^2.$$

Multivariate weighted reproducing kernel Hilbert space

Use the one-dimensional space as building block for d dimensions by taking weighted tensor products (tensor product basis):

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_{\mathfrak{u}\subseteq\{1,\dots,d\}} \gamma_{\mathfrak{u}} \prod_{j\in\mathfrak{u}} K(x_j,y_j) = \sum_{\boldsymbol{h}} \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} \frac{\phi_{h_j}(x_j)}{\sqrt{\lambda_{h_j}}} \frac{\overline{\phi_{h_j}(y_j)}}{\sqrt{\lambda_{h_j}}}$$

$$= \sum_{\boldsymbol{h}} r^{-2}_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})\, \phi_{\boldsymbol{h}}(\boldsymbol{x})\, \overline{\phi_{\boldsymbol{h}}(\boldsymbol{y})},$$

with

(You could now interpret α as the decay of the eigenvalues.)

$$r^{-2}_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h}) = \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j\in\mathfrak{u}} \lambda_{h_j}^{-1} = \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} \lambda_{h_j}^{-1},$$

and $\mathfrak{u}(\boldsymbol{h}) = \{h_j : h_j \neq 0\}$ the support of $\boldsymbol{h}$. Now, $\gamma_\emptyset = 1$, $Q_N(1) = 1$,

$$e^2(Q_N;\mathscr{H}) = -1 + \sum_{k,\ell=1}^{N} w_k\, w_\ell\, K(\boldsymbol{x}_k,\boldsymbol{y}_\ell).$$

Korobov type spaces $\mathscr{E}_{\alpha,p}$: Lattice rules

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^d} \widehat{f}(\boldsymbol{h}) \exp(2\pi \mathrm{i}\, \boldsymbol{h}\cdot\boldsymbol{x}), \quad \|f\|^p_{\mathscr{E}_{\alpha,p}} := \sum_{\boldsymbol{h}\in\mathbb{Z}^d} \left|\widehat{f}(\boldsymbol{h})\right|^p r^p_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h}) < \infty.$$

Traditional choice of r, for $\alpha > 1/q$:

$$r^p_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h}) = \gamma^{-1}_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} r^p_\alpha(h_j), \qquad r^p_\alpha(h_j) = \max(1, |h_j|^{p\alpha}).$$

- Unweighted ⇒ intractable. Sloan & Woźniakowski (2001).
- Strong tractability with product weights iff $\sum_{j\geq 1} \gamma_{\{j\}} < \infty$. Ditto.
- CBC construction of lattice rule achieving optimal $O(N^{-\alpha+\delta})$, $\delta > 0$, if $\sum_{j\geq 1} \gamma_{\{j\}}^{1/(2\alpha)} < \infty$ with constant independent of d for product weights. Kuo (2003).
- Unweighted strong tractability for sufficient large α and e.g. $r^2_\alpha(h_j) = (2\pi h)^{2\alpha}$, i.e., periodic Sobolev space, under the condition of permutation-invariance, e.g. $f(x_1, x_2) = f(x_2, x_1)$. N., Suryanarayana, Weimar (2015,201z).
- Several results on exponential convergence: Dick, Larcher, Pillichshammer, Woźniakowski (2011), Kritzer, Pillichshammer, Woźniakowski (2014), Irrgeher, Kritzer, Leobacher, Pillichshammer. (2015), Nguyen, N. (201x), Nguyen, N., Cools (201y), ...

Cosine space $\mathscr{C}_{\alpha,p}$: Tent-transformed lattice rules

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h}\in\mathbb{N}_0^d} \widehat{f}_{\cos}(\boldsymbol{h}) \prod_{\substack{j=1\\ h_j\neq 0}}^{d} \sqrt{2}\cos(\pi h_j x_j), \quad \|f\|_{\mathscr{C}_{\alpha,p}}^p := \sum_{\boldsymbol{h}\in\mathbb{N}_0^d} \left|\widehat{f}_{\cos}(\boldsymbol{h})\right|^p r_{\alpha,\boldsymbol{\gamma}}^p(\boldsymbol{h}).$$

In the Hilbert case, for $\alpha = 1$ and $d = 1$:

$$\langle f, g\rangle_{\mathscr{C}_{1,2}} = \left(\int_0^1 f(x)\,\mathrm{d}x\right)\left(\int_0^1 g(x)\,\mathrm{d}x\right) + \frac{(2\pi)^2}{\gamma}\int_0^1 f'(x)\,g'(x)\,\mathrm{d}x.$$

For $\gamma = (2\pi)^2$ this is the inner product of the unanchored Sobolev space $\mathscr{W}_{1,2}$, Dick, N., Pillichshammer (2013).

- This gives deterministic $O(N^{-1})$ for tent-transformed lattice rules in $\mathscr{W}_{1,2}$. Tent-transform: elementwise $x \mapsto 1 - |2x-1|$.
- Same tractability results as for Korobov space.
- Thus for product weights, optimal dimension-independent $O(N^{-\alpha+\delta})$, $\delta > 0$, if $\sum_{j\geq 1} \gamma_{\{j\}}^{1/(2\alpha)} < \infty$.

Sobolev spaces and Walsh spaces

- Unanchored Sobolev space $\mathcal{W}_{\alpha,p}$ of order $\alpha = 1$:
  - randomly shifted lattice rules/sequences, or
  - (deterministic) tent-transformed lattice rules/sequences.
- Walsh spaces based on Walsh expansion of function:
  - digital nets/sequences,
  - polynomial lattice rules.
- Higher order Sobolev spaces embedded in Walsh spaces:
  - higher order polynomial lattice rules,
  - interlaced digital nets/sequences,
  - interlaced polynomial lattice rules.

See, e.g., Hickernell (2002), Dick & Pillichshammer (2009), Baldeaux, Dick, Leobacher, N., Pillichshammer (2012), Dick, Kuo, Le Gia, N., Schwab (2014), ...

Polynomial lattice rules ≡ lattice rules over finite field

For $x \in [0,1)$ identify (uniquely) (Let's fix the base of the finite field to $b = 2$.)

$$x = \sum_{i \geq 1} x_i 2^{-i} \simeq (x_1 x_2 \cdots)_2 \simeq x_1 \chi^1 + x_2 \chi^2 + \cdots = x(\chi).$$

Likewise for $k \in \mathbb{N}_0$

$$k = \sum_{i \geq 0} k_i 2^i \simeq (\cdots k_2 k_1 k_0)_2 \simeq k_0 \chi^0 + k_1 \chi^1 + \cdots = k(\chi).$$

For a polynomial $P(\chi)$ over $\mathbb{Z}_2$ with $\deg(P) = n$ define the polynomial lattice points

$$\boldsymbol{x}_k(\chi) = \frac{\boldsymbol{z}(\chi) k(\chi) \bmod P(\chi)}{P(\chi)} \quad \text{over } \mathbb{Z}_2[\chi], \quad \text{for } k = 0, \ldots, b^m - 1.$$

The points in the unit cube $[0,1)^d$ are obtained by evaluating each $x(\chi)$ in $\chi = b = 2$ (up to χ^{-n}, or further).

Recap: Lattice rules

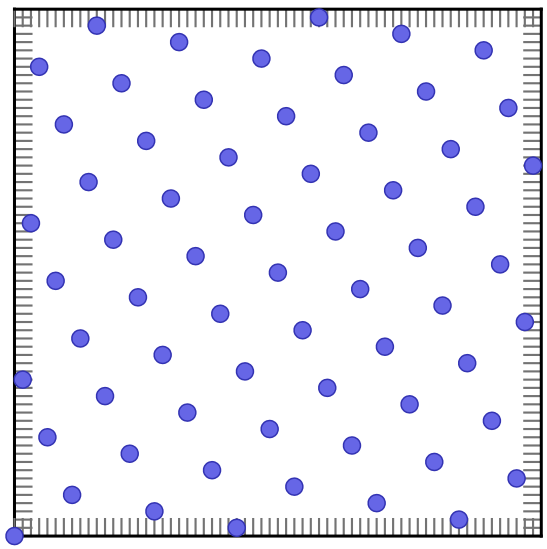
For a generating vector $\boldsymbol{z} \in \mathbb{Z}_N^d$:

$$\boldsymbol{x}_k = \frac{\boldsymbol{z}k \bmod N}{N}, \qquad \text{for } 0 \le k < N.$$

Then

$$Q_N(f; \boldsymbol{z}, N) = \frac{1}{N} \sum_{k=0}^{N-1} f(\boldsymbol{x}_k).$$

Components of z_j relatively prime to N:
then z_j defines a permutation for dimension j.



Polynomial lattice rules

For a generating vector $\boldsymbol{z}(\chi) \in (\mathbb{Z}_2[\chi]/P(\chi))^d$:

$$\boldsymbol{x}_k(\chi) = \frac{\boldsymbol{z}(\chi)k(\chi) \bmod P(\chi)}{P(\chi)}, \qquad \text{for } \deg(k(\chi)) < 2^m.$$

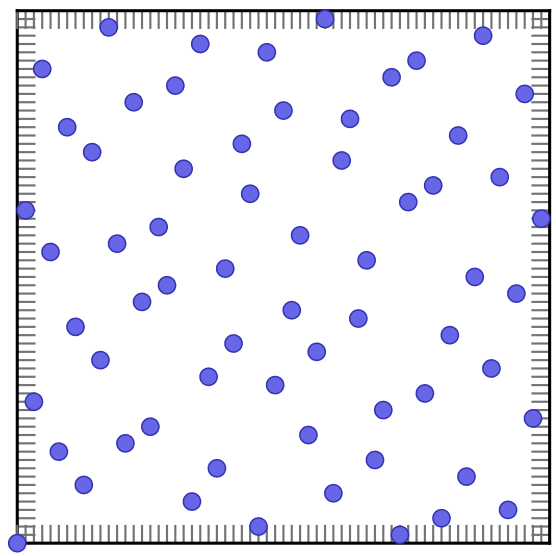
Then

$$Q_N(f; \boldsymbol{z}(\chi), P(\chi)) = \frac{1}{N} \sum_{k=0}^{2^m-1} f(\boldsymbol{x}_k(\chi)|_{\chi=2}).$$

If $z_j(\chi)$ relatively prime to $P(\chi)$:
then $z_j(\chi)$ defines a permutation for dim j.

→ Typically, take $P(\chi)$ irreducible over $\mathbb{Z}_2[\chi]$.

Niederreiter (1992), Dick & Pillichshammer (2009),
Lemieux & L'Écuyer (2002), ...



Digital net formulation

Assembling the n bits of each component of $\boldsymbol{x}_k$ in a vector $\vec{x}_{k,j}$, and the m bits of k in a vector $\vec{k}$, this is equivalent to

$$\vec{x}_{k,j} = C_j \vec{k} \qquad \text{over } \mathbb{Z}_2$$

where $C_j \in \mathbb{Z}_2^{n \times m}$ is a Hankel matrix formed by the recursion of $z_j(\chi)/P(\chi)$ over $\mathbb{Z}_2$. I.e., for each dimension j, we have

$$\begin{pmatrix} x_{k,j,1} \\ x_{k,j,2} \\ x_{k,j,3} \\ \vdots \\ x_{k,j,n} \end{pmatrix} = \begin{pmatrix} & & \\ \vec{c}_{j,1} & \cdots & \vec{c}_{j,m} \\ & & \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_{m-1} \end{pmatrix} = \bigoplus_{\substack{i=0 \\ k_i \neq 0}}^{m-1} \vec{c}_{j,i},$$

where $\oplus$ is addition over $\mathbb{Z}_2$.
$\rightarrow$ "In parallel" on n bits this is the `xor` operation...

Representation

By the same representation of integers and vectors over $\mathbb{Z}_2$:

$$C_j = \begin{pmatrix} \vec{c}_{j,1} & \cdots & \vec{c}_{j,m} \end{pmatrix} \in \mathbb{Z}_2^{n \times m} \quad \simeq \quad \begin{pmatrix} c_{j,1} & \cdots & c_{j,m} \end{pmatrix} \in \mathbb{Z}^m.$$

Inverse: Print generating matrices given as integers $\mathtt{C} = (C_{j,i})$, $j = 1, \ldots, d$, $i = 1, \ldots, m$:

```
n = max(floor(log2(C(:)))+1); [d, m] = size(C);
permute(reshape(rot90(dec2bin(C, n)-'0'), n, d, m),[1 3 2])
```

- Each matrix $C_j \in \mathbb{Z}_2^{n \times m}$ is represented as m integers.
- Integers need to have at least $\geq n$ bits.
- Use top row as least significant bit.
- Running through k in Gray code ordering only changes one bit.
- Next point: `xor` "state" with column of changed bit.

Interlaced points

Given a digital net in base b with $s = \alpha d$ dimensions:

$$\boldsymbol{x}_k = \begin{pmatrix} x_{k,1} \\ x_{k,2} \\ \vdots \\ x_{k,\alpha} \\ \hline x_{k,\alpha+1} \\ \vdots \\ x_{k,2\alpha} \\ \hline \vdots \\ x_{k,\alpha d} \end{pmatrix} = \begin{pmatrix} 0.x_{k,1,1}\, x_{k,1,2}\, x_{k,1,3} \cdots x_{k,1,m} \\ 0.x_{k,2,1}\, x_{k,2,2}\, x_{k,2,3} \cdots x_{k,2,m} \\ \vdots \\ 0.x_{k,\alpha,1}\, x_{k,\alpha,2}\, x_{k,\alpha,3} \cdots x_{k,\alpha,m} \\ \hline 0.x_{k,\alpha+1,1}\, x_{k,\alpha+1,2}\, x_{k,\alpha+1,3} \cdots x_{k,\alpha+1,m} \\ \vdots \\ 0.x_{k,2\alpha,1}\, x_{k,2\alpha,2}\, x_{k,2\alpha,3} \cdots x_{k,2\alpha,m} \\ \hline \vdots \\ 0.x_{k,\alpha d,1}\, x_{k,\alpha d,2}\, x_{k,\alpha d,3} \cdots x_{k,\alpha d,m} \end{pmatrix}$$

gives d dimensions with αm digits

$$y_{k,1} = 0.x_{k,1,1}\, x_{k,2,1} \cdots x_{k,\alpha,1} | x_{k,1,2}\, x_{k,2,2} \cdots x_{k,\alpha,2} | \cdots,$$

and so on...

See, e.g., Dick (2007), Dick & Goda (2013), ...

Interlaced points

Given a digital net in base b with $s = \alpha d$ dimensions:

$$\boldsymbol{x}_k = \begin{pmatrix} x_{k,1} \\ x_{k,2} \\ \vdots \\ x_{k,\alpha} \\ \hline x_{k,\alpha+1} \\ \vdots \\ x_{k,2\alpha} \\ \hline \vdots \\ x_{k,\alpha d} \end{pmatrix} = \begin{pmatrix} 0.x_{k,1,1}\, x_{k,1,2}\, x_{k,1,3} \cdots x_{k,1,m} \\ 0.x_{k,2,1}\, x_{k,2,2}\, x_{k,2,3} \cdots x_{k,2,m} \\ \vdots \\ 0.x_{k,\alpha,1}\, x_{k,\alpha,2}\, x_{k,\alpha,3} \cdots x_{k,\alpha,m} \\ \hline 0.x_{k,\alpha+1,1}\, x_{k,\alpha+1,2}\, x_{k,\alpha+1,3} \cdots x_{k,\alpha+1,m} \\ \vdots \\ 0.x_{k,2\alpha,1}\, x_{k,2\alpha,2}\, x_{k,2\alpha,3} \cdots x_{k,2\alpha,m} \\ \hline \vdots \\ 0.x_{k,\alpha d,1}\, x_{k,\alpha d,2}\, x_{k,\alpha d,3} \cdots x_{k,\alpha d,m} \end{pmatrix}$$

gives d dimensions with αm digits

$$y_{k,1} = 0.x_{k,1,1}\, x_{k,2,1} \cdots x_{k,\alpha,1} | x_{k,1,2}\, x_{k,2,2} \cdots x_{k,\alpha,2} | \cdots,$$

and so on...

See, e.g., Dick (2007), Dick & Goda (2013), ...

Interlaced points

Given a digital net in base b with $s = \alpha d$ dimensions:

$$\boldsymbol{x}_k = \begin{pmatrix} x_{k,1} \\ x_{k,2} \\ \vdots \\ x_{k,\alpha} \\ \hline x_{k,\alpha+1} \\ \vdots \\ x_{k,2\alpha} \\ \hline \vdots \\ x_{k,\alpha d} \end{pmatrix} = \begin{pmatrix} 0.x_{k,1,1}\, x_{k,1,2}\, x_{k,1,3} \cdots x_{k,1,m} \\ 0.x_{k,2,1}\, x_{k,2,2}\, x_{k,2,3} \cdots x_{k,2,m} \\ \vdots \\ 0.x_{k,\alpha,1}\, x_{k,\alpha,2}\, x_{k,\alpha,3} \cdots x_{k,\alpha,m} \\ \hline 0.x_{k,\alpha+1,1}\, x_{k,\alpha+1,2}\, x_{k,\alpha+1,3} \cdots x_{k,\alpha+1,m} \\ \vdots \\ 0.x_{k,2\alpha,1}\, x_{k,2\alpha,2}\, x_{k,2\alpha,3} \cdots x_{k,2\alpha,m} \\ \hline \vdots \\ 0.x_{k,\alpha d,1}\, x_{k,\alpha d,2}\, x_{k,\alpha d,3} \cdots x_{k,\alpha d,m} \end{pmatrix}$$

gives d dimensions with αm digits

$$y_{k,1} = 0.x_{k,1,1}\, x_{k,2,1} \cdots x_{k,\alpha,1} | x_{k,1,2}\, x_{k,2,2} \cdots x_{k,\alpha,2} | \cdots,$$

and so on...

See, e.g., Dick (2007), Dick & Goda (2013), ...

Generate the points

- Generate points in Gray code ordering.
- Only one integer (column) needs to be `xor`'d.
- Need to know which bit changed...

de Bruijn sequence for bit change:

```
db = [0, 1, 28, 2, 29, 14, 24, 3, 30, 22, 20, 15, 25, 17, 4, 8,
      31, 27, 13, 23, 21, 19, 16, 7, 26, 12, 18, 6, 11, 5, 10, 9];
jc = @(v) db( bitshift (mod(double(bitand(int32(v),-int32(v)))*0x077CB531,
                pow2(32)),-27)+1)+1; % mind the double because of Matlab madne
```

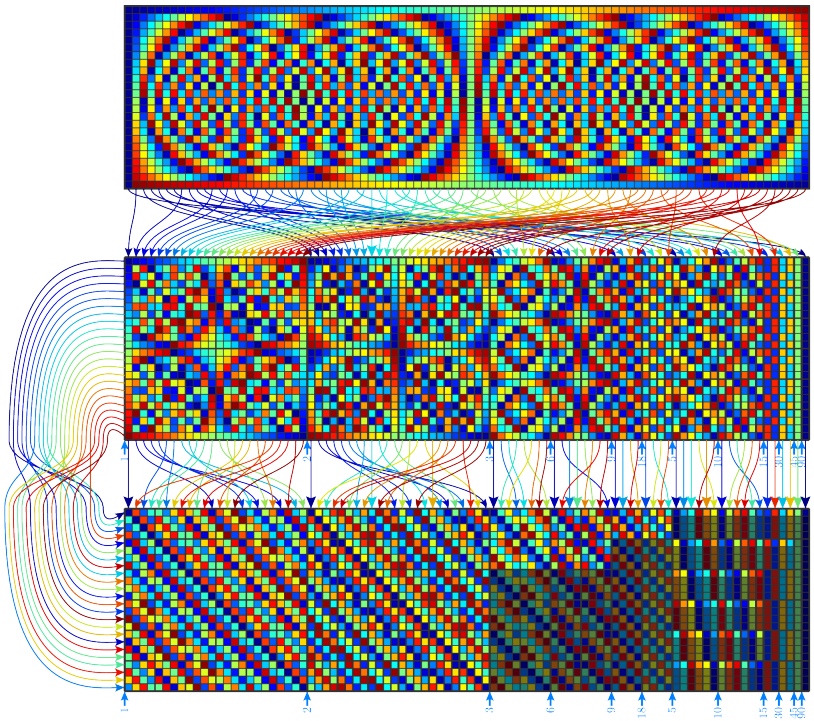
Generate points (for the fun of it):

```
k = 1:pow2(m)-1; K = numel(k);
P = reshape(reshape(mod(cumsum(
        reshape( fliplr (dec2bin(C(:,jc(k)),n)-'0'),d,K,n), 2), 2),
        d*K, n) * pow2(-1:-1:-n)', d, K);
```

→ Don't be silly: use `digitalseq_b2g` instead.

Construction of lattice rules and polynomial lattice rules

Point sets constructed for weighted spaces using fast component-by-component constructions using number theoretic transforms.



See https://www.cs.kuleuven.be/~dirkn/qmc4pde/ and https://www.cs.kuleuven.be/~dirkn/fast-cbc/.

See, e.g., N. & Cools (2006a,2006b), Cools, Kuo, & N. (2006), Dick, Kuo, Le Gia, N. & Schwab (2014), N. (2014), Kuo & N. (2016), ...
Variations and speedups by: Gantner, Kritzer, Laimer, Leobacher, Pillichshammer, Schwab, ...

Point generators

- Matlab/Octave: procedural generators like Matlab's `rand`:
  - `latticeseq_b2.m`: radical inverse lattice sequence generator,
  - `digitalseq_b2g.m`: gray coded radical inverse digital sequence generator (incl. higher-order, max 53 bit).
- Python: iterator classes, which can be used as standalone point generators from the command line (`__main__`):
  - `latticeseq_b2.py`: iterator based (`__iter__`), `set_state` for parallel computing,
  - `digitalseq_b2g.py`: ditto, arbitrary precision using `mpmath` if needed.
- C++: header file based implementation with driver program for the command line:
  - `latticeseq_b2.(h|cpp)`: complies to `ForwardIterator` concept, `set_state` for parallel computing,
  - `digitalseq_b2g.(h|cpp)`: ditto, max 64 bit.

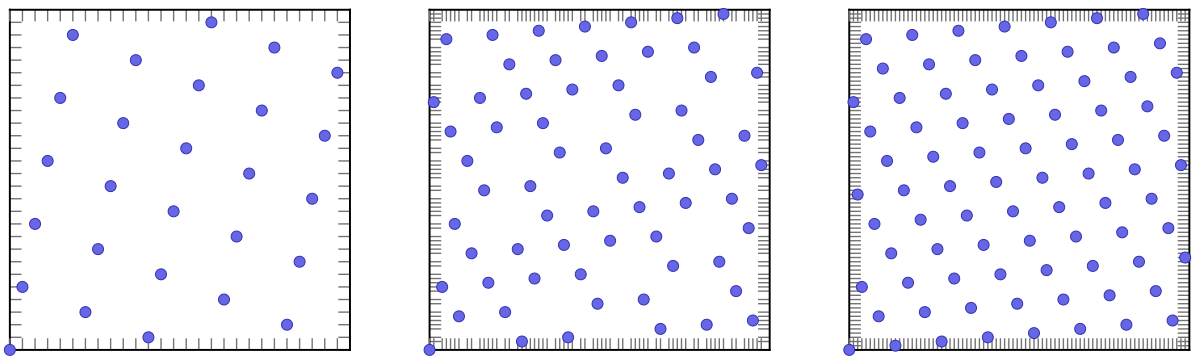
See https://www.cs.kuleuven.be/~dirkn/qmc4pde/ and https://www.cs.kuleuven.be/~dirkn/qmc-generators/.

Welcome to The Magic Point Shop!

Different flavours of quasi-Monte Carlo points to choose:

- Lattice rules.
- Lattice sequences.
- Polynomial lattice rules.
- Interlaced Sobol' sequences (higher-order).
- Interlaced polynomial lattice rules (higher-order).
- Interlaced Niederreiter–Xing sequence (higher-order).

And code (C++ and Matlab) to use them...



Subsidiaries: QMC4PDE: construct points for parametrised PDEs.

Bounds on derivatives

Assume bounds on derivatives of f, $\nu \in \{0,\ldots,\alpha\}^d$, up to $|\nu| = \sum_{j=1}^{s} \nu_j \leq \alpha$

$$\left\|(\partial^{\nu} f)(x_1,\ldots,x_d)\right\|_{\infty} = O\Big((|\nu + a_1|!)^{d_1} \prod_{j=1}^{d} (a_2\,\beta_j)^{\nu_j}\Big).$$

"SPOD-type" (smoothness-driven product and order dependent decay) and "POD-type" (product and order dependent decay, $\alpha = 1$) — coming from PDE context, see, e.g., Dick, Kuo, Le Gia, N., Schwab (2014), Kuo & N. (2016).

With e.g.

$$\beta_j = c\, j^{-d_2}$$

for some $d_2 > d_1 \geq 0$ and $c > 0$.
The parameter d_2 is the "decay".

Construction of specialized (polynomial) lattice rules

Lattice rules:

- randomly shifted lattice rules on $[0,1]^s$
  $\rightarrow$ randomly shifted Sobolev space W_1: $O(N^{-1})$
- randomly shifted lattice rules constructed for $\mathbb{R}^s$
  $\rightarrow$ "unbounded integrand space": $O(N^{-1})$
- tent-transformed lattice rules
  $\rightarrow$ Sobolev space W_1: $O(N^{-1})$

Digital nets:

- randomly shifted polynomial lattice rules
  $\rightarrow$ randomly shifted Sobolev space: $O(N^{-1})$
- higher order polynomial lattice rules
  $\rightarrow$ Sobolev space W_r embedded in ho Walsh space: $O(N^{-r})$
- higher order interlaced polynomial lattice rules
  $\rightarrow$ Sobolev space W_r embedded in ho Walsh space: $O(N^{-r})$

Take home message

- You can use QMC!
- It's easy!
- You are not afraid of the function spaces!
- You gladly found out about higher-order convergence $N^{-\alpha}$!
- You are happily finding online software resources!
- You walk out happy to the coffee break...

This talk was mainly based on (self-promote):

- N. (2014): review of fast construction,
- Kuo & N. (2016): QMC4PDE review,
- Magic Point website (and fast CBC construction scripts),
- QMC4PDE website.

But of course there is also L'Écuyer & Munger (TOMS, 2016), Gantner & Schwab's website, Kuo's website.